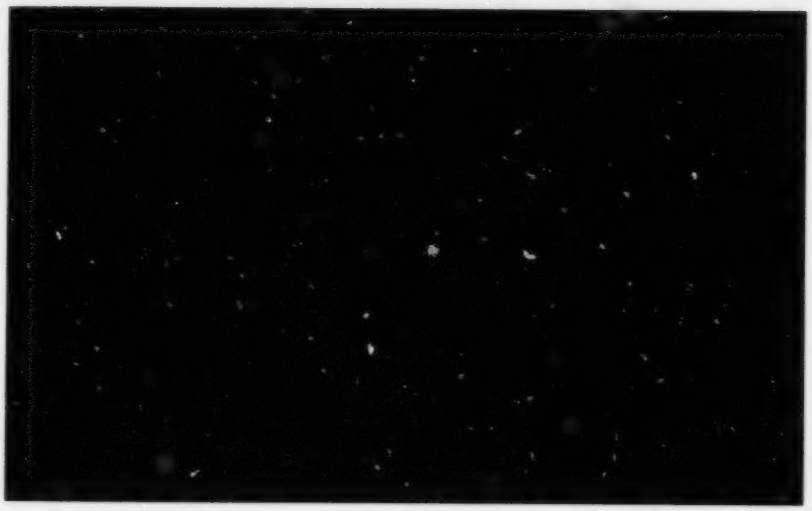
CENTRE DE RECHERCHE
SUR LES TRANSPORTS
(C.R.T.)
CENTRE FOR RESEARCH
ON TRANSPORTATION

UNIVERSITÉ DE MONTRÉAL
ÉCOLE DES HAUTES ÉTUDES COMMERCIALES
ÉCOLE POLYTECHNIQUE



CENTRE DE RECHERCHE SUR LES TRANSPORTS
UNIVERSITÉ DE MONTRÉAL
C.P. 6128, SUCCURSALE A.
MONTRÉAL
CANADA H3C 3J7

TÉLÉPHONE: (514) 343-7575
TÉLEX: (21) 05-561325
EMAIL: crt@crt.umontreal.ca
TÉLÉCOPIEUR (FAX): (514) 343-7121

91-02546/

CRT-755

Location of Facilities on a Network with Edge Failures

by
Horst A. Eiselt[1]
Michel Gendreau[2]
and
Gilbert Laporte[2]

[1] Faculty of Administration, University of New Brunswick, P.O. Box 4400, Fredericton, New Brunswick, Canada E3B 5A3.

[2] Centre de recherche sur les transports, Université de Montréal, C.P. 6128, succursale A, Montréal, Canada H3C 3J7.

Centre de recherche sur les transports — Publication #755

January 1991

Dépôt légal – 1er trimestre 1991

ABSTRACT

In this paper, the following location problem is analyzed. Let $N = (V, E)$ be an undirected connected simple network, where V is the vertex set, $|V| = n$, and E is the edge set. There is a non-negative demand w_j associated with every vertex v_j. It is assumed that every edge (v_i, v_j) has a probability of failure p_{ij}, and that failures can never occur on two edges simultaneously. The problem consists of locating p facilities on the network so that the total expected demand disconnected from the facilities is minimized. This problem occurs naturally in the fields of computer and telecommunications networks. A number of important results are proved. First, there always exists a solution in which all facilities are located at vertices. Second, the problem can always be solved optimally on the so-called leaf-tree associated with the network. Third, when $p = 1$, the problem is a 1-median problem. When $p > 1$, there always exists an optimal solution for which all facilities are located at pendent vertices of the tree. Finally, when $p > 2$, the problem with $p + 1$ facilities can be solved in a greedy fashion, starting from a solution to the problem with p facilities. An exact algorithm for this problem is described. It can be executed in either $O(np + |E|)$ time, or in $O(n \log n + |E|)$ time. A numerical example is provided.

Keywords: stochastic facility location problems, networks, trees, leaf-trees, edge failures, reliability.

RÉSUMÉ

Dans cet article, on analyse le problème de localisation suivant. Soit $N = (V, E)$ un réseau non dirigé, connexe et simple, où V est l'ensemble des sommets, $|V| = n$, et E l'ensemble des arêtes. On associe une demande non négative w_j à chaque sommet v_j. Chaque arête (v_i, v_j) peut tomber en panne avec une probabilité p_{ij}, et la probabilité de deux pannes simultanées est nulle. Le problème consiste à localiser p installations sur le réseau, de façon à minimiser la demande totale espérée isolée de ces installations. Ce problème possède des applications dans le domaine des réseaux d'ordinateurs et de télécommunication. On démontre un certain nombre de résultats pour ce problème. D'abord, il existe toujours une solution dans laquelle toutes les installations sont localisées à des sommets du réseau. Deuxièmement, on peut toujours résoudre le problème sur un arbre associé au réseau. Troisièmement, lorsque $p = 1$, le problème est un problème de 1-médiane. Lorsque $p > 1$, il existe toujours une solution optimale pour laquelle toutes les installations sont localisées aux feuilles de l'arbre. Finalement, lorsque $p > 2$, le problème avec $p + 1$ installations se résout de façon gloutonne, à partir d'une solution du problème avec p installations. On décrit un algorithme exact pour ce problème. Cet algorithme peut être exécuté en temps $O(np + |E|)$, ou en temps $O(n \log n + |E|)$. Un exemple numérique illustre l'algorithme.

Mots-clés: problèmes de localisation stochastiques, réseaux, arbres, pannes sur les arêtes, fiabilité.

1. Introduction

In this paper, we consider a location problem on a network with possible edge failures. More precisely, define an undirected connected simple network $N = (V,E)$, where $V = \{v_1, ..., v_n\}$ is the vertex set, $E = \{(v_i, v_j),\ v_i, v_j \epsilon V\}$ is the edge set, and $(v_i, v_j) \equiv (v_j, v_i)$. A positive weight or demand w_j is associated with every vertex v_j. The probability of failure of edge (v_i, v_j) is denoted by p_{ij}, and we restrict our attention to situations where the probability of simultaneous failure on more than one edge is negligeable. The problem consists of locating p facilities on the network so as to minimize the total expected demand disconnected from these facilities.

Applications of this problem occur naturally in the fields of computer and telecommunications networks. Nel and Colbourn [17] study the location of a broadcast facility in a network with multiple potential edge failures. However, our assumption of a single potential failure is often justified in practice, as shown in a recent study by Sansó, Soumis and Gendreau [20] on the reliability of telecommunications networks.

Related but different stochastic location problems have been studied by several authors. A number of papers are concerned with median problems with random demands or edge lengths (Frank [9], Louveaux [13], Louveaux and Thisse [14], Mirchandani and Odoni [16], Weaver and Church [22]). A problem similar to ours, in which failures occur at facilities rather than on edges, has been investigated by Drezner [8]. ReVelle [18] and ReVelle and Hogan [19] have considered stochastic covering

problems arising, for example, in the location of emergency facilities. Finally, several studies have addressed location queueing problems (see, for example, Berman, Larson and Chiu [2], Chiu [6], Chiu, Berman and Larson [7], Brandeau and Chiu [4]). Additional references on stochastic location problems are provided in Brandeau and Chiu [3], and in Berman, Chiu, Larson, Odoni and Batta [1].

Our problem is also related to the deterministic p-median problem (see, e.g., Mirchandani [15]), which consists of locating p facilities on a network in order to minimize the total weighted distance between vertices and their closest facility. It is well known that the p-median problem on general networks is NP-hard when $p > 1$; however, when N is a tree, it can be solved in $O(n^2p^2)$ time (Kariv and Hakimi [12]). The 1-median problem can be solved in $O(n^3)$ time on a general network (Hakimi [11]) and in $O(n)$ time on a tree (Goldman [10]). Now consider the following scenario where p facilities are first located and every demand is allocated to a facility through a determined chain. In the event of edge failure, all demands flowing through that edge are lost. The problem of minimizing total expected lost demand can then be solved as a p-median problem in which the length of edge (v_i, v_j) is p_{ij} (recall that only one edge failure is allowed). Our problem is similar, except that *recourse* is now possible, i.e. upon failure, demands may be rerouted to an open facility as long as a chain exists. In other words, only those demands isolated in a subnetwork containing no facility are lost. It will be shown that this case is simpler than a p-median problem and can in fact be solved in polynomial time. In contrast, the more general problem of

locating p facilities in a network where several edge failures can occur simultaneously is NP-hard, even when $p = 1$ (Nel and Colbourn [17]).

Because of the relationship of our problem with the p-median problem and since only one edge failure can occur, for the sake of convenience, we call it the "simple stochastic p-median problem" and use the abbreviation SSMP.

The remainder of the paper is organized as follows. In Section 2, some fundamental properties of the SSMP are derived. These serve as the basis for an exact algorithm described in Section 3, and illustrated in Section 4.

2. Fundamental Properties

Hakimi [11] has shown that the p-median problem always possesses an optimal solution in which all facilities are located at vertices. This property also holds for our problem.

Proposition 1. There always exists an SSMP solution in which all facilities are located at vertices.

Proof. Consider a solution for which at least one of the facilities is located on edge (v_i, v_j). Relocating the facility to either v_i or v_j does not deteriorate the current solution. Indeed, if (v_i, v_j) is not the failing edge, all demand previously allocated to that facility can still be served. If (v_i, v_j) fails and the facility is not at v_i or v_j, the demand allocated to this facility may be lost. Lost demand can never increase

by moving the facility to either v_i or v_j. □

A further simplification can be obtained by solving the SSMP on the so-called *leaf-network* of *N* defined as follows (see, e.g., Carré [5], p. 157). In an undirected network *N*, an edge (v_i, v_j) is called a ***bridge*** if its removal disconnects v_i and v_j. Removing all bridges from *N* yields k subnetworks or *leaves* $N_1, ..., N_k$ containing no bridges. The *leaf-network* of *N* is a network *N' = (V',E')* where *V'* contains one vertex for each leaf of *N*, and edge (v_i, v_j) belongs to *E'* if and only if *E* contains a bridge between leaves N_i and N_j of *N* – see Figure 1. Note that if *N* is connected, then *N'* is a tree.

Locating a facility at any vertex of a leaf of *N* ensures that in case of edge failure, all vertices of the leaf remain connected to the facility since any leaf is 2-connected. Hence the precise location of a facility in a leaf is immaterial and we can state:

Proposition 2. An optimal SSMP solution on a network *N* can be obtained by solving an SSMP on its leaf-network *N'*. □

Tarjan [21] proposes an algorithm that determines the leaf-network of a connected network *N = (V,E)* in $O(|E|)$ time, provided *N* is represented by a list structure. From now on, it will be assumed that the SSMP is always solved on a tree *T = (V,E)*. We will consider the two cases $p = 1$ and $p \geq 2$ separately.

If $p = 1$, no recourse is possible in case of failure and therefore the SSMP can be solved as a 1-median problem, in $O(n)$ time, as in the no recourse case.

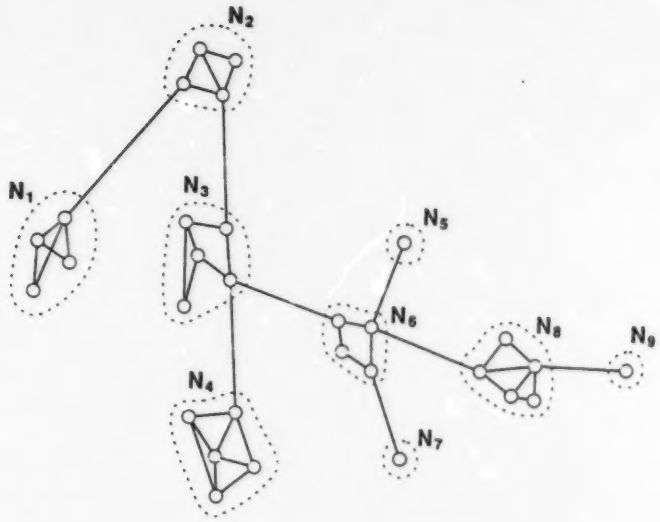


Figure 1a. Network showing leaves and bridges

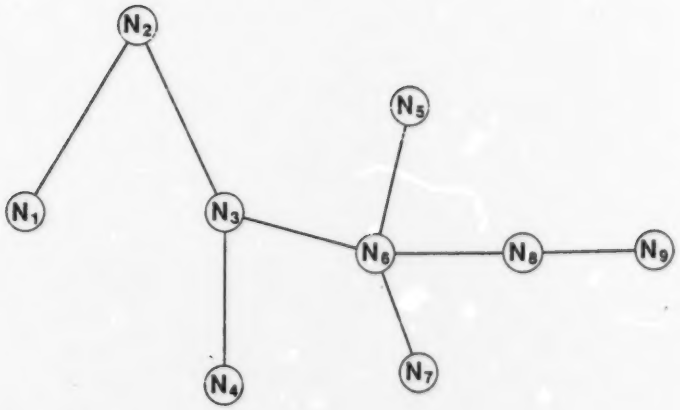


Figure 1b. Corresponding leaf-network

When $p = 1$, any edge failure causes at least one vertex to be disconnected from the facility. However, when $p \geq 2$, this is not the case of all edges and it is useful to introduce the notion of *critical* edges. These are the edges whose deletion causes some vertices to be disconnected from the facilities. More formally, critical edges are defined as follows. In a tree, deleting any edge (v_i, v_j) yields two subtrees rooted at v_i and v_j, respectively. Denote by $T_i^j = \left(V_i^j, E_i^j\right)$ the subtree rooted at v_i when (v_i, v_j) is deleted. (The subscript denotes the root of the subtree and the combination of the subscript and the superscript refers to the deleted edge.) For any given solution, edge (v_i, v_j) is critical if all facilities are located in only one of T_i^j or T_j^i. Letting C be the set of critical edges, and assuming that for any critical edge (v_i, v_j) the subtree T_j^i contains no facility, the SSMP objective function can then be expressed simply as

$$z = \sum_{(v_i, v_j) \in C} p_{ij}\, w(T_j^i),$$

where $w\left(T_j^i\right) = \sum_{v_k \in V_j^i} w_k$.

Now consider two facilities located at distinct vertices v_i and v_j. All vertices (and the corresponding demands) on the unique chain $(v_i, ..., v_j)$ are *protected* in the sense that any edge failure on the tree leaves these vertices connected to either v_i or v_j. Moreover, all vertices of the tree are also protected from failures of (non-critical) edges on the chain $(v_i, ..., v_j)$. Another observation is that if the number p of facilities is at least equal to the number r of pendent vertices of the tree, there exists an optimal solution with only r facilities located at the pendent vertices, since all chains are then

protected. We now prove that a similar property holds when $p < r$.

Proposition 3. Consider an SSMP defined on a tree with r pendent vertices, and assume $2 \leq p < r$. Then there exists an optimal solution in which all facilities are located at distinct pendent vertices of the tree.

Proof. Consider a solution containing a facility located at a non-pendent vertex v_i. Let $T^i_{i_1}, ..., T^i_{i_q}$ be the q subtrees obtained by removing from T vertex v_i and its incident edges $(v_i, v_{i_1}), ..., (v_i, v_{i_q})$. If each of these subtrees already contains a facility, there is no loss of optimality in moving the facility located at v_i to any adjacent vertex v_{i_h}, since (v_i, v_{i_h}) is non-critical. If at least one subtree $T^i_{i_h}$ contains no facility, then no vertex of $T^i_{i_h}$ is protected. Moving the facility from v_i to its neighbour v_{i_h} in $T^i_{i_h}$ makes edge (v_i, v_{i_h}) non-critical and also protects v_{i_h} without affecting the status of any other edge or vertex, and without deteriorating the solution. This process can be repeated until all p facilities are located at pendent vertices. □

In the following, we will show that when $p \geq 3$, the SSMP can be solved by means of a greedy algorithm. Denote by SSMP(p) the problem with p facilities. Suppose $p \geq 2$ and that an optimal solution to SSMP(p) is known. Then locating an additional facility optimally with respect to the already located facilities (i.e. solving the conditional SSMP) provides an optimal solution to SSMP($p+1$). More formally, we prove:

Proposition 4. Let $2 \leq p < r$, and denote by S_p an optimal solution to SSMP(p).

Then, there exists an optimal solution S_{p+1} to SSMP$(p+1)$ such that $S_p \subseteq S_{p+1}$.

Proof. For any $S \subseteq V$, denote by $z(S)$ the value of a solution with one facility located at each vertex of S. Let S'_{p+1} be an optimal solution to SSMP$(p+1)$ for which all facilities are located at distinct pendent vertices. We will first show that there exists a vertex $\tilde{v} \epsilon S'_{p+1}$ such that $z\left(S'_{p+1}\right) = z(S_p \cup \{\tilde{v}\})$.

Without loss of generality, define S'_{p+1} as $\{v_1, ..., v_{p+1}\}$. Let V'_i $(i = 1, ..., p+1)$ be the sets of vertices of V which are protected by $S'_{p+1} \backslash \{v_i\}$, and $V' = \bigcap_{i=1}^{p+1} V'_i$. In other words, V' is the set of vertices protected by any p out of the $p+1$ facilities in S'_{p+1}. Note that V' contains no pendent vertices. Let $\{T_1, ..., T_q\}$ $(q \geq p+1)$ be the forest obtained by removing from T the subtree spanned by the vertices of V', and all other edges incident to these vertices – see Figure 2, and observe that any T_k may contain at most one vertex of S'_{p+1}. It follows that there exists at least one T_k containing a vertex $\tilde{v} \epsilon S'_{p+1}$ and no vertex of S_p. Now let T'_p be the subtree spanned by S_p, and T'_{p+1} be the subtree spanned by $S'_{p+1} \backslash \{\tilde{v}\}$ – see Figure 3. Neither T'_p nor T'_{p+1} shares any vertex with the subtree T_k containing $\tilde{v}$. Also let E_p be the set of edges on the chain from $\tilde{v}$ to T'_p and E_{p+1}, the set of edges on the chain from $\tilde{v}$ to T'_{p+1}, and note that $E_{p+1} \subseteq E_p$. All edges of E_p are critical with respect to S_p, and non-critical with respect to $S_p \cup \{\tilde{v}\}$. Similarly, edges of E_{p+1} are critical with respect to $S'_{p+1} \backslash \{\tilde{v}\}$, and non-critical with respect to S'_{p+1}. It follows that

$$z\left(S'_{p+1} \backslash \{\tilde{v}\}\right) - z\left(S'_{p+1}\right) \leq z(S_p) - z(S_p \cup \{\tilde{v}\}). \tag{1}$$

But $z(S_p) \leq z\left(S'_{p+1} \backslash \{\tilde{v}\}\right)$ since S_p is optimal for SSMP(p). Therefore $z\left(S'_{p+1}\right) \geq z(S_p \cup \{\tilde{v}\})$. The equality holds since S'_{p+1} is optimal for SSMP($p+1$). Hence, any solution $\{v\}$ to the conditional SSMP with respect to S_p will also satisfy $z(S_p \cup \{v\}) = z\left(S'_{p+1}\right)$, and thus S_{p+1} can be defined as $S_p \cup \{v\}$. □

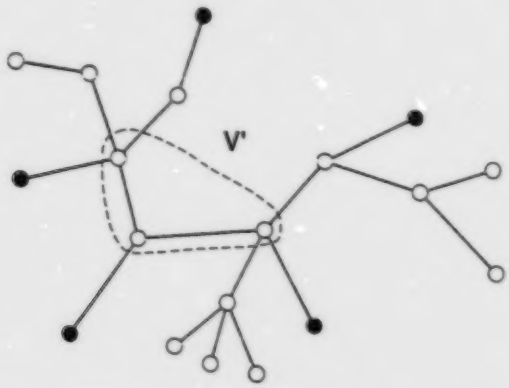


Figure 2a. Network with already located facilities (black nodes)

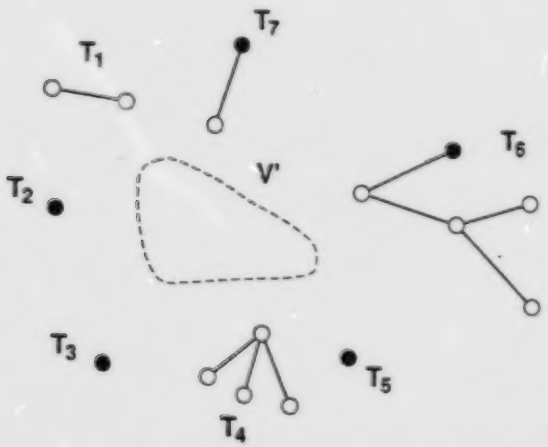
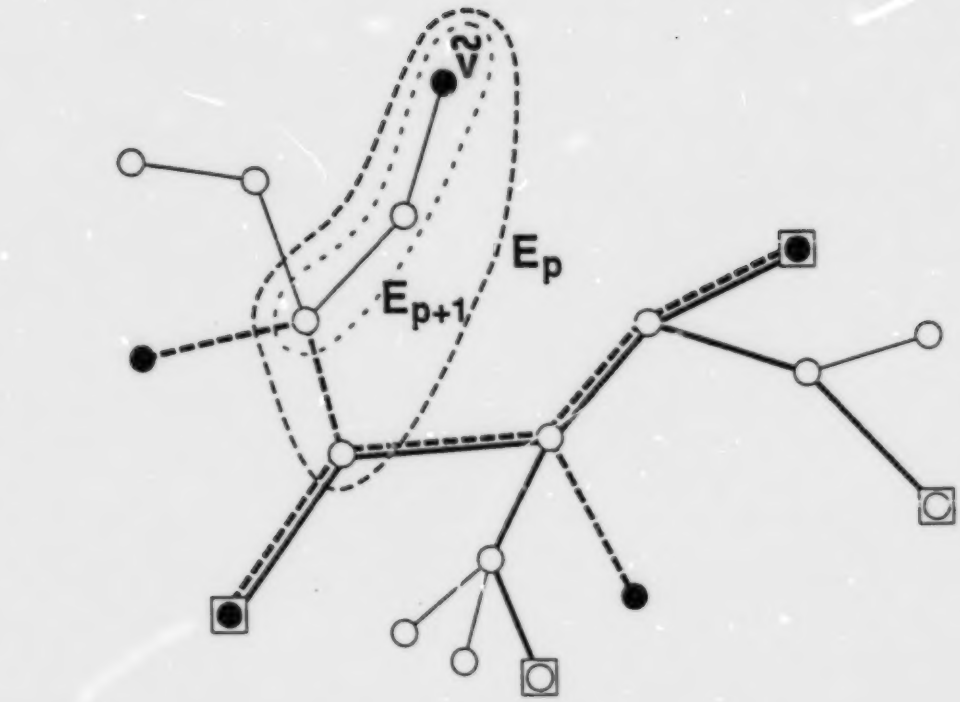


Figure 2b. Forest obtained by removing V' and incident edges



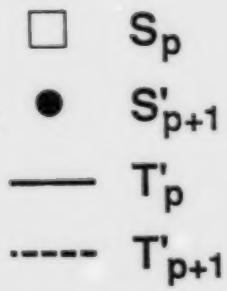


Figure 3. Network showing E_p and E_{p+1}

3. Algorithm

In this section we describe an $O(np)$ algorithm for determining the optimal SSMP solution when $2 \leq p < r$. We also show how this algorithm can be modified to yield an $O(n \log n)$ procedure, if desired. The algorithm first locates two facilities optimally. Then the remaining facilities are located by solving a sequence of conditional problems (see Proposition 4).

We first consider the problem of optimally locating a facility within a subtree. This will lead to a procedure (LOCATE_IN_SUBTREE) central to the overall algorithm. Given an edge (v_i, v_j) with no facility in T_j^i and at least one facility in T_i^j, determine the optimal location of a new facility in T_j^i. In order to solve this problem, first define $\alpha\left(T_j^i\right)$ as the expected additional demand that is protected if a facility is located optimally in T_j^i. The value of $\alpha\left(T_j^i\right)$ can be computed recursively as follows:

$$\alpha(T_j^i) = p_{ij}\, w(T_j^i) + max_{(v_j,v_k)\epsilon E_j^i} \left\{\alpha\left(T_k^j\right)\right\}, \qquad (1)$$

where the second term is zero if T_j^i consists of a single vertex. The computation of $\alpha\left(T_j^i\right)$ is done backwards, starting from the pendent vertices of T_j^i. The optimal location v^* is then determined by the sequence of maximizing subtrees in (1), starting from vertex v_j, and finishing when the pendent vertex v^* is encountered. The complexity of this procedure is $O\left(\left|V_j^i\right|\right)$ time.

Since our algorithm requires first locating two facilities, it will be useful to derive

some additional technical properties of SSMP(2) solutions. Consider a vertex v_i of degree q and the subtrees $T^i_{i_1} = (V^i_{i_1}, E^i_{i_1}), ..., T^i_{i_q} = \left(V^i_{i_q}, E^i_{i_q}\right)$ obtained by removing all edges incident to v_i, and ordered so that $\alpha(T^i_{i_1}) \geq \ ... \geq \alpha\left(T^i_{i_q}\right)$.

Proposition 5. For any $v_i \epsilon V$, there exists an optimal solution to SSMP(2) in which at least one of the facilities is located in $T^i_{i_1}$.

Proof. If the two facilities are located in two different subtrees $T^i_{i_h}$ and $T^i_{i_l}$, it follows from the definition of α that it is optimal to take $k = 1$ and $\ell = 2$. If the two facilities are located in the same subtree $T^i_{i_h}$, there exists a (unique) vertex v_s in $T^i_{i_h}$ such that the facilities are located in two different subtrees $T^s_{s_1}$ and $T^s_{s_2}$ of $T^i_{i_h}$. Since $\alpha(T^s_{s_1}) \leq \alpha(T^i_{i_h})$ and $\alpha(T^s_{s_2}) \leq \alpha(T^i_{i_h})$, it follows that $\alpha(T^s_{s_1}) + \alpha(T^s_{s_2}) \leq 2\alpha(T^i_{i_h}) \leq \alpha(T^i_{i_1}) + \alpha(T^i_{i_2})$, unless $h = 1$. Finally, if one facility is located at v_i, it follows again from the definition of α that the other facility must be located in $T^i_{i_1}$. □

Proposition 6. Let $\{v', v''\}$ be an optimal solution to SSMP(2). Then there exists an edge (v_i, v_j) on the chain $(v', ..., v'')$ such that v_i is the root of $T^j_{j_1}$ and v_j is the root of $T^i_{i_1}$. Such an edge is called a *beam* (as in a balance).

Proof. For notational simplicity relabel the vertices of the chain $(v', ..., v'')$ as $(v_1 = v', v_2, ..., v_k = v'')$. Proposition 5 implies that for every vertex v_ℓ of that chain with $2 \leq \ell \leq k-1$, either $v_{\ell-1} \epsilon V^\ell_{\ell_1}$ and $v_{\ell+1} \epsilon V^\ell_{\ell_2}$, or $v_{\ell-1} \epsilon V^\ell_{\ell_2}$ and $v_{\ell+1} \epsilon V^\ell_{\ell_1}$. Assign to every such vertex v_ℓ a left-label of 1 if $v_{\ell-1} \epsilon V^\ell_{\ell_1}$, and a left-label of 2 otherwise.

Similarly, assign right-labels to every v_ℓ. Since facilities are located at v_1 and v_k, vertex v_1 can now be assigned a right-label of 1, and v_k a left-label of 1. Each of the $k-1$ edges of the chain $(v_1, ..., v_k)$ receives two labels through this procedure. Since k labels are equal to 1 and $k-2$ labels are equal to 2, there must exist an edge with two labels equal to 1. This edge is a beam. □

Proposition 5 implies that there exists an optimal SSMP(2) solution with one facility on either side of the beam. It follows that SSMP(2) is easily solved once a beam is known. The next proposition will prove useful in identifying a beam.

Proposition 7. Let v_i be a non-pendent vertex and v', an optimal location at a pendent vertex of $T^i_{i_1}$. Then there exists a beam on the chain $(v', ..., v_i)$.

Proof. Let v'' be a pendent vertex such that $\{v', v''\}$ is an optimal solution to SSMP(2). The two chains $(v', ..., v_i)$ and $(v', ..., v'')$ have a maximal common subchain $(v', ..., v_s)$. Consider the two labels of v_s on the chain $(v', ..., v_{s-1}, v_s, v_{s+1}, ..., v'')$. As in Proposition 6, these labels are equal to 1 and 2. Furthermore, since v' is optimally located in $T^i_{i_1}$ by selecting at each step a subtree for which α is maximum, we must have $\alpha(T^s_{s-1}) \geq \alpha(T^s_{s+1})$ and the left-label of v_s may be set equal to 1. Thus, using the same argument as in the proof of Proposition 6, there must be a beam on $(v', ..., v_s)$. □

We are now in a position to describe an exact algorithm for SSMP(p) when $2 \leq p < r$. In Phase 1, values of α are computed with respect to an arbitrary

non-pendent *pivot vertex* v_i. Then, in Phase 2, a beam is identified in $T^i_{i_1}$, while the α values on the path from v_i to the beam are updated. Two facilities are then optimally located on each side of the beam, using the procedure described at the beginning of this section. In Phase 3, if $p > 2$, the remaining $p - 2$ facilities are located sequentially, in a greedy fashion. Finally, the value of the objective function is computed in Phase 4.

The following notation will be used:

S: the solution set, i.e. the set of vertices at which facilities are located;

V'': the set of unprotected vertices, given S ($V'' = V \backslash V'$ where V' is the set of protected vertices – see Section 2);

$d(v_j)$: the degree of v_j;

V^j: the set of neighbours of v_j.

In addition, at any stage of the algorithm, the pivot vertex v_i is known. The role of this vertex is to give an orientation to the tree and to associate a specific subtree with any given vertex v_j. More precisely, the subtree $T_j = (V_j, E_j)$ associated with v_j is the subtree rooted at v_j when the chain $(v_i, ..., v_j)$ is removed. Hence, we can define

$\alpha(v_j) = \alpha(T_j)$;

$\beta(v_j) = w(T_j) = \sum_{v_k \in V_j} w_k$;

$f(v_j)$: the father vertex of v_j (on the chain $(v_i, ..., v_j)$).

In fact, the algorithm uses $\alpha_1(v_j) \equiv \alpha(v_j)$, and $\alpha_2(v_j)$ computed as $\alpha(v_j)$ (see equation (1)), except that the "maximum" is replaced by the "second maximum". The roots of the subtrees yielding $\alpha_1(v_j)$ and $\alpha_2(v_j)$ are called $j_1(v_j)$ and $j_2(v_j)$, respectively.

It is also important to note that as long as v_i is the pivot, $\alpha(T_i)$ is meaningless and $\alpha_1(v_i)$ and $\alpha_2(v_i)$ are simply used to store the first and second maximum of $\alpha(v_j)$ over the neighbours v_j of v_i. These are later used to identify the beam.

In the labeling step of the algorithm, labels are assigned to vertices starting first from the pendent vertices, and working towards the pivot v_i. The final label values (with respect to v_i) of a vertex v_j may be determined once all vertices in $V_j \backslash \{v_j\}$ have received their labels. To guarantee that vertices are processed in a proper sequence, we maintain throughout the labeling step a list P of vertices eligible for labeling. In Step 1 of the algorithm, P is defined as the list of pendent vertices. During the labeling process, a vertex $v_j \neq v_i$ becomes eligible for labeling and can thus be added to P when all its neighbours but one (its father) have been labeled. This situation is detected by decrementing the degree of v_j by 1 whenever one of its neighbours is labeled, and by testing whether $d(v_j) = 1$ after every such update.

In the backtracking step of the algorithm, v_j is always selected as the root of subtree $T^i_{i_1}$. To identify a beam, it is thus sufficient to find a pair of vertices (v_i, v_j) such that v_i is the root of subtree $T^j_{j_1}$ (see Proposition 6). At the beginning of Step 3,

$\alpha_1(v_i)$ is set equal to $\alpha\left(T_i^j\right)$ and $\alpha_1(v_j)$ becomes equal to the maximum of $\alpha\left(T_j^k\right)$ over all other neighbours v_k of v_j. Therefore, (v_i, v_j) is a beam if and only if $\alpha_1(v_i) \geq \alpha_1(v_j)$.

ALGORITHM SSMP

Phase 1 (α Computation).

Step 0 (Initialization).

Using edge data, for every vertex $v_j \epsilon V$, create V^j, and compute $d(v_j)$;

set $\alpha_1(v_j) := \alpha_2(v_j) := j_1(v_j) := j_2(v_j) := 0$ and $\beta(v_j) := w_j$.

Compute $W := \sum_{v_j \epsilon V} w_j$.

Step 1 (Creation of pendent vertex list).

Set P:=Ø.

For every $v_j \epsilon V$ **do**

if $d(v_j) = 1$, **then** $P := P \cup \{v_j\}$.

Step 2 (Labeling).

Select arbitrarily $v_i \epsilon V \backslash P$.

While $P \neq \emptyset$, **do**

select arbitrarily $v_k \epsilon P$, and let v_j be the (only remaining) vertex in V^k;

set $f(v_k) := v_j$;

< finish computing the best and second-best $\alpha(v_k)$ >

$\alpha_1(v_k) := \alpha_1(v_k) + p_{kj}\ \beta(v_k)$

$\alpha_2(v_k) := \alpha_2(v_k) + p_{kj}\ \beta(v_k)$

< update the α values for v_j >

if $\alpha_1(v_k) > \alpha_1(v_j)$

then $\alpha_2(v_j) := \alpha_1(v_j)$, $j_2(v_j) := j_1(v_j)$

$\alpha_1(v_j) := \alpha_1(v_k)$, $j_1(v_j) := v_k$;

else if $\alpha_1(v_k) > \alpha_2(v_j)$

then $\alpha_2(v_j) := \alpha_1(v_k)$, $j_2(v_j) := v_k$;

add $\beta(v_k)$ to $\beta(v_j)$;

delete v_k from V^j and update $d(v_j)$;

if $d(v_j) = 1$ **and** $v_j \neq v_i$ **then** add v_j to P.

Phase 2 (Solving SSMP(2)).

Step 3 (Backtracking and beam identification).

$v_j := j_1(v_i)$

$\beta(v_i) := W - \beta(v_j)$

$\alpha_1(v_i) := \alpha_2(v_i) + p_{ij}\ \beta(v_i)$, $j_1(v_i) := j_2(v_i)$

$\alpha_1(v_j) := \alpha_1(v_j) - p_{ij}\ \beta(v_j)$

$\alpha_2(v_j) := \alpha_2(v_j) - p_{ij}\ \beta(v_j)$

if $\alpha_1(v_i) < \alpha_1(v_j)$

then if $\alpha_1(v_i) > \alpha_2(v_j)$

then $\alpha_2(v_j) := \alpha_1(v_i)$, $j_2(v_j) := v_i$

$f(v_i) := v_j$

$v_i := v_j$

go to Step 3.

$< (v_i, v_j)$ **is a beam** $>$

Step 4 (Locating the first two facilities).

Set $V'' := V$ and $S := \emptyset$;

perform procedure LOCATE_IN_SUBTREE (v_i);

perform procedure LOCATE_IN_SUBTREE (v_j).

Phase 3 (Locating the remaining facilities if $p > 2$).

Step 5.

For $t := 3$ **to** p **do**

select $v^* \epsilon V''$ such that $f(v^*) \notin V''$, maximizing $\alpha_1(v)$;

perform procedure LOCATE_IN_SUBTREE (v^*).

Phase 4 (Computing the objective function). [Optional].

Step 6.

Set $z := 0$.

For $v_k \epsilon V''$ **do**

$v_j := f(v_k)$

$z := z + p_{jk}\,\beta(v_k)$. □

Procedure LOCATE_IN_SUBTREE (root)

$v_s :=$ root

while $j_1(v_s) \neq 0$ **do**

delete v_s from V'',

$v_s := j_1(v_s)$;

delete v_s from V'',

add v_s to S. □

Since Step 5 requires $O(np)$ operations, and every other step can be executed in $O(n)$ time, solving SSMP(p) on a tree requires $O(np)$ time. However, algorithm SSMP can be modified to run in $O(n \log n)$ time. As it is now, Step 5 is the only step which requires more than $O(n)$ operations. This is because for each new facility, we have to determine $v^* \epsilon V''$ with $f(v^*) \notin V''$ maximizing $\alpha_1(v)$. By maintaining dynamically the set $\overline{V}$ of vertices v in V'' with $f(v) \notin V''$ sorted in non-increasing order of $\alpha_1(v)$ value (this can be implemented using a heap structure), the selection of v^* can now be performed in constant time. On the other hand, the overall complexity of the p executions of procedure LOCATE_IN_SUBTREE will now be $O(n \log n)$. To verify this, note that each vertex will be added and deleted from $\overline{V}$ at most once and that each such operation requires at most $O(\log n)$ time if $\overline{V}$ is a heap since

there can never be more than n elements in $\overline{V}$.

The modified versions of Step 4, Step 5 and LOCATE_IN_SUBTREE are as follows. In these, for every $v_k \epsilon V$, $\overline{V}^k$ denotes the set of sons of v_k, i.e. $\overline{V}^k = \{v \epsilon V : (v, v_k) \epsilon E \text{ and } v \neq f(v_k)\}$.

Modified Step 4.

Set $V'' := V$ and $S := \emptyset$;

for every $v_k \epsilon V$ **do** $\overline{V}^k := \emptyset$;

for every $v_k \epsilon V \backslash \{v_i, v_j\}$ **do**

let $v_s := f(v_k)$,

add v_k to $\overline{V}^s$;

Set $\overline{V} := \{v_i, v_j\}$;

Modified Step 5.

For $t := 3$ **to** p **do**

select v^* the top element of $\overline{V}$;

perform procedure LOCATE_IN_SUBTREE (v^*).

Modified Procedure LOCATE_IN_SUBTREE (root)

$v_s :=$ root

while $j_1(v_s) \neq 0$ **do**

delete v_s from V'' and $\overline{V}$,

add $\overline{V}^s$ to $\overline{V}$,

$v_s := j_1(v_s)$;

delete v_s from V'' and $\overline{V}$,

add v_s to S. □

Solving SSMP(p) on a general network $N=(V,E)$ can therefore be accomplished in $O(np + |E|)$ time or in $O(n \log n + |E|)$ time, when Tarjan's algorithm is used to construct the leaf-network of N (provided appropriate data structures are used in the implementation).

4. Numerical Example

We now illustrate the $O(np)$ version of the algorithm on a problem with $n = 10$ and $p = 4$ (see Figure 4). For simplicity, vertex v_j will be represented by its index j.

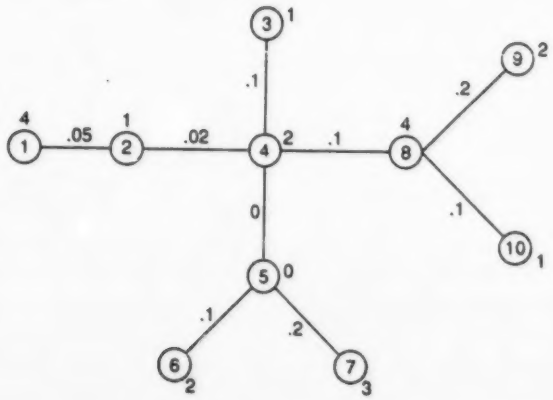


Figure 4. Tree network with vertex demands and probabilities of edge failures

In Step 0, all labels are initialized: $\alpha_1(v_j) := \alpha_2(v_j) := j_1(v_j) := j_2(v_j) := 0$, and $\beta(v_j) := w_j$ for all j. The sum of all weights is $W := 20$. In Step 1, P is defined as the set of pendent vertices, i.e. $P := \{1, 3, 6, 7, 9, 10\}$. In Step 2, vertex 2 is arbitrarily selected as the pivot. We first compute the labels of all pendent vertices. Recall that $f(v_j)$ is the father of v_j and hence, $f(1) := 2$, $f(3) := 4$, $f(6) := 5$, $f(7) := 5$, $f(9) := 8$, and $f(10) := 8$. Also, for any pendent vertex v_j, we have $\alpha_1(v_j) := \alpha_2(v_j) := p_{j,f(v_j)}\, w_j$; $j_1(v_j) := j_2(v_j) := 0$ and $\beta(v_j) := w_j$. Hence, $\alpha_1(1) := \alpha_2(1) := 0.05 \times 4 = 0.2$, $\alpha_1(3) := \alpha_2(3) := 0.1 \times 1 = 0.1$, $\alpha_1(6) := \alpha_2(6) := 0.1 \times 2 = 0.2$, $\alpha_1(7) := \alpha_2(7) := 0.2 \times 3 = 0.6$, $\alpha_1(9) := \alpha_2(9) := 0.02 \times 2 = 0.4$, $\alpha_1(10) := \alpha_2(10) := 0.1 \times 1 = 0.1$. As these vertices are processed, labels are also assigned to their fathers: $\beta(2) := w_2 + \beta(1) = 1 + 4 = 5$, $\alpha_1(2) := \alpha_1(1) := 0.2$, $j_1(2) := 1$, $\alpha_2(2) := 0$, and $j_2(2) := 0$ (since vertex 2 has only one son processed so far). Similarly, $\beta(5) := w_5 + \beta(6) + \beta(7) = 0 + 2 + 3 = 5$, $\alpha_1(5) := \max\{\alpha_1(6) = 0.2, \alpha_1(7) = 0.6\} = 0.6$, hence, $j_1(5) := 7$, $\alpha_2(5) := 0.2$ and $j_2(5) := 6$. In the same way, we have $\beta(4) := 3$, $\alpha_1(4) := 0.1$, $j_1(4) := 3$, $\alpha_2(4) := 0$, and $j_2(4) := 0$; $\beta(8) := 7$, $\alpha_1(8) := 0.4$, $j_1(8) := 9$, $\alpha_2(8) := 0.1$, and $j_2(8) := 10$. After these operations, the set P is equal to {5, 8} (note that the pivot vertex never belongs to P). The labels of these vertices are now either computed or updated. Starting with vertex 8, we have $f(8) := 4$, $\alpha_1(8) := \alpha_1(8) + p_{8,4}\, \beta(8) = 0.4 + 0.1 \times 7 = 1.1$, $\alpha_2(8) := \alpha_2(8) + p_{8,4}\, \beta(8) = 0.1 + 0.1 \times 7 = 0.8$. The labels of vertex 4 are updated: $\beta(4) := \beta(4) + \beta(8) = 3 + 7 = 10$; since $\alpha_1(8) > \alpha_1(4)$,

we set $\alpha_1(4) := \alpha_1(8) := 1.1$, $j_1(4) := 8$, $\alpha_2(4)$ takes the old value of $\alpha_1(4)$, namely 0.1, and $j_2(4) := 3$. Similarly, the α labels of vertex 5 are recomputed as $\alpha_1(5) := 0.6$ and $\alpha_2(5) := 0.2$. The father vertex of 5 is identified as 4 and hence, the labels of vertex 4 are once more updated as $\beta(4) := 15$, $\alpha_1(4)$ and $j_1(4)$ remain unchanged since $\alpha_1(5) < \alpha_1(4)$; $\alpha_2(4) := \alpha_1(5) := 0.6$, and $j_2(4) := 5$ since $\alpha_1(5)$ was greater than $\alpha_2(4)$. At this stage, $P = \{4\}$, and the labels of vertex 4 are updated, together with those of its father, vertex 2: $\alpha_1(4) := 1.4$, $\alpha_2(4) := 0.9$, $\beta(2) := 20$, $\alpha_1(2) := 1.4$, $j_1(2) := 4$, $\alpha_2(2) := 0.2$, and $j_2(2) := 1$. The labeling process is now complete and the label values at the end of Phase 1 are summarized in Table 1.

Table 1. Label values at the end of Phase 1

Vertex v_j	$f(v_j)$	$\beta(v_j)$	$\alpha_1(v_j)$	$\alpha_2(v_j)$	$j_1(v_j)$	$j_2(v_j)$
1	2	4	0.2	0.2	0	0
2	-	20	1.4	0.2	4	1
3	4	1	0.1	0.1	0	0
4	2	15	1.4	0.9	8	5
5	4	5	0.6	0.2	7	6
6	5	2	0.2	0.2	0	0
7	5	3	0.6	0.6	0	0
8	4	7	1.1	0.8	9	10
9	8	2	0.4	0.4	0	0
10	8	1	0.1	0.1	0	0

In Phase 2, backtracking is executed from the pivot vertex 2 to $j_1(2) = 4$. Accordingly, the labels of these two vertices are updated as follows: $\beta(2) :=$

$W - \beta(4) = 20 - 15 = 5$, $\alpha_1(2) := \alpha_2(2) + p_{2,4}\,\beta(2) = 0.2 + 0.02 \times 5 = 0.3$, $j_1(2) := j_2(2) = 1$ ($\alpha_2(2)$ and $j_2(2)$ are no longer needed and are not updated); $\alpha_1(4) := \alpha_1(4) - p_{2,4}\,\beta(4) = 1.4 - 0.02 \times 15 = 1.1$, $\alpha_2(4) := \alpha_2(4) - p_{2,4}\,\beta(4) = 0.9 - 0.02 \times 15 = 0.6$. (Note that vertex 4 has regained the labels it had before it was last processed in Phase 1.) Since $\alpha_1(2) < \alpha_1(4)$ and $\alpha_1(2) < \alpha_2(4)$, we set $f(2) := 4$, $v_i := 4$ and the backtracking continues from vertex 4 to vertex $j_1(4) = 8$. Hence, we obtain $\beta(4) := 13$, $\alpha_1(4) := 1.9$, $j_1(4) := 5$, $\alpha_1(8) := 0.4$, $\alpha_2(8) := 0.1$. Since $\alpha_1(4) > \alpha_1(8)$, the beam (4, 8) has now been identified. The new label values at the end of Step 3 are summarized in Table 2.

Table 2. Labels of vertices 2, 4 and 8 at the end of Step 3

Vertex v_j	$f(v_j)$	$\beta(v_j)$	$\alpha_1(v_j)$	$\alpha_2(v_j)$	$j_1(v_j)$	$j_2(v_j)$
2	4	5	0.3	0.2	1	1
4	2	13	1.9	0.6	5	5
8	4	7	0.4	0.1	9	10

In Step 4, two facilities will be located, one in T_4^8, and one in T_8^4, using Procedure LOCATE_IN_SUBTREE. In the first case, backtracking is done from the root (vertex 4) to $j_1(4) = 5$, to $j_1(5) = 7$, which is a pendent vertex and thus, the location of the first facility. Similarly, the optimal location of the second facility in T_8^4 is vertex 9. At this stage the set of unprotected vertices is $V'' = \{1, 2, 3, 6, 10\}$.

In Phase 3, the two remaining facilities are located as follows. The set of vertices $v \epsilon V''$ such that $f(v) \notin V''$ is {2, 3, 6, 10}. The vertex v^* for which $\alpha_1(v)$ is

maximized is 2. The third facility is therefore located in $T_2^{f(2)}$, at vertex 1. The set V'' is now {3, 6, 10}, v^* is identified as vertex 6, and the fourth facility is located at that vertex, since it is pendent.

Finally in Phase 4, the value of the objective function can be computed. Since $V'' = \{3, 10\}$, it follows that $z := p_{f(3),3}\beta(3) + p_{f(10),10}\beta(10) = 0.1 \times 1 + 0.1 \times 1 = 0.2$.

ACKNOWLEDGEMENTS

This research was in part supported by the Canadian Natural Sciences and Engineering Research Council (grants OGP0009160, OGP0038816 and OGP0039682), and by a Quebec–New-Brunswick interprovincial research grant. This support is gratefully acknowledged. Thanks are also due to Erhan Erkut for valuable suggestions.

REFERENCES

[1] Berman, O., S.S. Chiu, R.C. Larson, A.R. Odoni and R. Batta, "Location of mobile units in a stochastic environment", in P.B. Mirchandani and R.L. Francis eds., *Discrete Location Theory*, Wiley, New York, 503–549 (1990).

[2] Berman, O., R.C. Larson and S.S. Chiu, "Optimal server location on a network operating as an M/G/1 queue", *Operations Research* 33, 746–770 (1985).

[3] Brandeau, M.L. and S.S. Chiu, "An overview of representative problems in location research", *Management Science* 35, 645–674 (1989).

[4] Brandeau, M.L. and S.S. Chiu, "The stochastic queue median trajectory in a plane with generalized distances", *Operations Research* (1991). Forthcoming.

[5] Carré, B., *Graphs and Networks*, Clarendon Press, Oxford (1979).

[6] Chiu, S.S., "Optimal M/G/1 server location on a tree network with continuous link demands", *Computers & Operations Research* 13, 653–669 (1986).

[7] Chiu, S.S., O. Berman and R.C. Larson, "Locating a mobile server queueing facility on a tree network", *Management Science* 31, 764–772 (1985).

[8] Drezner, Z., "Location of unreliable facilities", Working paper, California State University at Fullerton (1985).

[9] Frank, H., "Optimum locations on a graph with probabilistic demands", *Operations Research* 14, 409–421 (1966).

[10] Goldman, A.J., "Optimal center location in simple networks", *Transportation Science* 5, 212–221 (1971).

[11] Hakimi, S.L., "Optimal locations of switching centers and the absolute centers and medians of a graph", *Operations Research* 12, 450–459 (1964).

[12] Kariv, O. and S.L. Hakimi, "An algorithmic approach to network location problems. Part 2. The P-medians", *SIAM Journal on Applied Mathematics* 37, 539–560 (1979).

[13] Louveaux, F.V., "Discrete stochastic location models", in J.P. Osleeb and S.J. Ratick, eds., *Location Decisions: Methodology and Applications, Annals of Operations Research* 6, 23–34 (1986).

[14] Louveaux, F.V. and J.-F. Thisse, "Production and location on a network under demand uncertainty", *Operations Research Letters* 4, 145–149 (1985).

[15] Mirchandani, P.B., "The p-median problem and generalizations", in P.B. Mirchandani and R.L. Francis, eds., *Discrete Location Theory*, Wiley, New York, 55–117 (1990).

[16] Mirchandani, P.B. and A.R. Odoni, "Location of medians on stochastic networks", *Transportation Science* 13, 85–97 (1979).

[17] Nel, L.D. and C.J. Colbourn, "Locating a broadcast facility in an unreliable network", *INFOR* 28, 363–379 (1990).

[18] ReVelle, C., "Review, extension and prediction in emergency service siting models", *European Journal of Operational Research* 40, 58–69 (1989).

[19] ReVelle, C. and K. Hogan, "The maximum reliability location problem and α-reliable p-center problem: derivatives of the probabilistic location set covering problem", in F.V. Louveaux, M. Labbé and J.-F. Thisse, eds., *Facility Location Analysis: Theory and Applications, Annals of Operations Research* 18, 155–173 (1989).

[20] Sansó, B., F. Soumis and M. Gendreau, "On the evaluation of telecommunication network reliability using routing models", *IEEE Transactions on Communications* (1991). Forthcoming.

[21] Tarjan, E., "A note on finding the bridges of a graph", *Information Processing Letters* 2, 160–161 (1974).

[22] Weaver, J.R. and R.L. Church, "Computational procedures for location problems on stochastic networks", *Transportation Science* 17, 168–180 (1983).